Efficient PageRank Tracking in Evolving Networks

Naoto Ohsaka (UTokyo)
Takanori Maehara (Shizuoka University)
Ken-ichi Kawarabayashi (NII)

ERATO Kawarabayashi Large Graph Project
Background

Personalized PageRank Tracking

Applications

Search engine [Brin-Page. '98]
Spam detection [Chung-Toyoda-Kitsuregawa. ’09, ’10]

Growth of real networks

<table>
<thead>
<tr>
<th></th>
<th>Size</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>WWW</td>
<td>60T</td>
<td>600K pages / s</td>
</tr>
<tr>
<td>Twitter</td>
<td>300M</td>
<td>5K tweets / s</td>
</tr>
<tr>
<td>Google+</td>
<td>700M</td>
<td>+19 users / s</td>
</tr>
</tbody>
</table>
### Background

**Existing Work for PageRank Tracking**

<table>
<thead>
<tr>
<th>Method</th>
<th>m random edge insertions</th>
<th>Scalability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregation/Disaggregation</td>
<td>$O(m</td>
<td>S</td>
</tr>
<tr>
<td>[Chien et al. ’04]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Monte-Carlo</td>
<td>$O(m + \log m /\epsilon^2)$</td>
<td>68M edges</td>
</tr>
<tr>
<td>[Bahmani et al. ’10]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power method</td>
<td>$O(m^2 \log 1/\epsilon)$</td>
<td>11M edges</td>
</tr>
<tr>
<td>naive method</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Scalability**
  - Update time < 0.1s
  - Error $\approx 10^{-9}$
## Our Contribution

Propose a **simple, efficient, & accurate** method for Personalized PageRank tracking in evolving networks

<table>
<thead>
<tr>
<th></th>
<th>$m$ random edge insertions</th>
<th>Scalability</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>This work</strong></td>
<td>Ave. $\downarrow$ Max. out-deg $O(m + \Delta \log m / \epsilon)$</td>
<td>$3,700M$ edges</td>
</tr>
<tr>
<td>Aggregation/Disaggregation [Chien et al. '04]</td>
<td>$O(m</td>
<td>S</td>
</tr>
<tr>
<td><strong>Monte-Carlo</strong> [Bahmani et al. '10]</td>
<td>$O(m + \log m / \epsilon^2)$</td>
<td>$68M$ edges</td>
</tr>
<tr>
<td><strong>Power method</strong> naive method</td>
<td>$O(m^2 \log 1/\epsilon)$</td>
<td>$11M$ edges</td>
</tr>
</tbody>
</table>

**Scalability**
- Update time < 0.1s
- Error $\approx 10^{-9}$
Preliminaries

Definition of Personalized PageRank

- Linear equation
  A solution $x$ of
  $$x = \alpha Px + (1 - \alpha)b$$
  - Preference vector
  - Transition matrix
  - Decay factor = 0.85

- Random walk modeling web browsing
  - Moves to a random out-neighbor w.p. $\alpha$
  - Jumps to a random vertex w.p. $1 - \alpha$
  (biased by $b$)
Preliminaries

Computing PageRank in Static Graphs

- Solving eq. \( x = \alpha Px + (1 - \alpha)b \)
  - Power method \( x^{(v)} = \alpha Px^{(v-1)} + (1 - \alpha)b \)
  - Gauss-Seidel [Del Corso-Gullí-Romani. Internet Math.'05]
  - LU/QR factorization [Fujiwara-Nakatsuji-Onizuka-Kitsuregawa. VLDB’12]
  - Krylov subspace method [Maehara-Akiba-Iwata-Kawarabayashi. VLDB’14]

- Estimating the frequency \( x_v \) of visiting \( v \)
  - Monte-Carlo simulation
Preliminaries

Tracking PageRank in Evolving Graphs

- **Aggregation/disaggregation**
  
  [Chien-Dwork-Kumar-Simon-Sivakumar. Internet Math.'04]
  
  Apply the power method to a subgraph

- **Monte-Carlo**

  [Bahmani-Chowdhury. VLDB’10]
  
  Maintain & update random-walk segments

\[ \Omega \left( \frac{1}{\epsilon^2} \right) \] Too many
Proposed Method

Problem Setting

**Given** \( G(0), \alpha, b \) at time 0

\[
G(0) = (V, E(0))
\]

\[
G(t - 1) = (V, E(t - 1))
\]

\[
G(t) = (V, E(t))
\]

\( G(0) \) to \( G(t) \) represents the evolution of the graph from time 0 to time \( t \).

**Problem** at time 0

Compute approx. PPR \( x(0) \) of \( G(0) \)

\[ \| x(0) - x^*(0) \|_\infty < \epsilon \]

**Problem** at time \( t \)

Compute approx. PPR \( x(t) \) of \( G(t) \)

**Given at time** \( t \):

Edges inserted to/deleted from

\[
\square \quad \text{Edges inserted to/deleted from}
\]
Proposed Method

The Idea

Solving eq. \( x(t) = \alpha P(t)x(t) + (1 - \alpha)b \)

1. \( x(t - 1) \) is a **GOOD** initial solution for \( x(t) \)
2. Improving approximate solution **locally**

- We use the **Gauss-Southwell** method 🤪 [Southwell. ‘40, ’46]
  
  *a.k.a. Bookmark coloring algorithm* [Berkhin. Internet Math.’06]
  
  **Local algorithm**
  [Spielman-Teng. SIAM J. Comput.’13] [Andersen-Chung-Lang. FOCS’06]
Proposed Method

Gauss-Southwell Method [Southwell. ’40,’46]

- **$\nu$th solution** $x^{(\nu)}$
- **$\nu$th residual** $r^{(\nu)} = (1 - \alpha)b - (I - \alpha P)x^{(\nu)}$

Goal: $r^{(\nu)} \rightarrow 0$

$v = 1,2,3, ...$

$i \leftarrow$ a vertex with largest $|r_i^{(\nu-1)}|$

If $|r_i^{(\nu-1)}| < \epsilon$ terminate

Update $x^{(\nu-1)}$ & $r^{(\nu-1)}$ locally so that $r_i^{(\nu)} = 0$

### # iterations

Stops within $\|r^{(0)}\| / (1 - \alpha)\epsilon$ iter.

$\|r^{(\nu)}\| \leq \|r^{(\nu-1)}\| - (1 - \alpha)\epsilon$

### Accuracy

$\|x^* - x^{(\nu)}\|_\infty \leq \frac{\epsilon}{1 - \alpha}$

$\sqrt{r_i^{(\nu)} < \epsilon}$
At time $t$:

\[
x(t)^{(0)} = x(t - 1) \\
r(t)^{(0)} = r(t - 1) + \alpha(P(t) - P(t - 1))x(t - 1)
\]

Apply the Gauss-Southwell method
At time $t$:
\[
x(t)^{(0)} = x(t - 1)
\]
\[
r(t)^{(0)} = r(t - 1) + \alpha (P(t) - P(t - 1)) x(t - 1)
\]

Apply the Gauss-Southwell method

Increase of $\|r\|_1$

Computation time of $\boxed{\text{Max. out-deg \uparrow}} = \mathcal{O}(\Delta \times \#\text{iters.})$

depends on $\| \|_1$ How small?
Consider any change including full construction

\[ G(t - 1) \rightarrow G(t) \]

Increase of \[ \|r\|_1 \leq 2\alpha \]

Same as static computation
Proposed Method

Performance Analysis: Random Edge Insertion

A single-edge is randomly inserted for each time

\[ G(0) \xrightarrow{\text{Insert an edge}} G(1) \xrightarrow{\text{...}} G(m) \]

\[ E(0) = \emptyset \]

Lemma 3 in [Bahmani-Chowdhury. VLDB’10]

Monte-Carlo

[Bahmani-Chowdhury. VLDB’10]

\[ \mathbb{E}[^{\# \text{ updated seg.}}] = O(R \log m) \]

\[ R = \Omega(1/\varepsilon^2) \] is total # seg.

Our method

\[ \mathbb{E}[^{\text{increase of } \|r\|_1}] \leq 2\alpha/t \]
Proposed Method

Performance Analysis: Results

Our result for random edge insertion (Prop. 6 in the paper)

If $m$ edges are randomly and sequentially inserted,
expected total $\# \text{iter.}$ is $\Theta(\log m / \epsilon)$

$\Rightarrow$ expected total time is $\Theta(m + \Delta \log m / \epsilon)$

Our result for any change (Prop. 5 in the paper)

$\# \text{iter.}$ for any change is amortized $\Theta(1/\epsilon)$

$\Rightarrow$ Time is amortized $\Theta(\Delta/\epsilon)$
Experiments

Setting: Single-edge Insertion

- Parameter settings
  - $\alpha = 0.85$
  - $b$ has 100 non-zero elements
  - $\epsilon = 10^{-9}$

$G(0)$  $\rightarrow$  $G(1)$  $\rightarrow$  $\ldots$  $\rightarrow$  $G(10^5)$

- Except the last 100,000 edges
- Add an edge chronologically or randomly
- The whole graph
## Experiments

### Efficiency Comparison: Time for an Edge Insertion

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$</td>
<td>V</td>
<td>=1M$</td>
<td>$</td>
</tr>
<tr>
<td></td>
<td>$</td>
<td>E</td>
<td>=5M$</td>
<td>$</td>
</tr>
<tr>
<td><strong>This work</strong></td>
<td>7 μs</td>
<td>77 μs</td>
<td>29,383 μs</td>
<td>2 μs</td>
</tr>
<tr>
<td>Aggregation/Disaggregation [Chien et al. ’04]</td>
<td>320 μs</td>
<td>40,336 μs</td>
<td>&gt;100,000 μs</td>
<td>&gt;100,000 μs</td>
</tr>
<tr>
<td>Monte-Carlo [Bahmani et al. ’10]</td>
<td>444 μs</td>
<td>9,196 μs</td>
<td>&gt;100,000 μs</td>
<td>&gt;100,000 μs</td>
</tr>
<tr>
<td>Warm start power method</td>
<td>80,994 μs</td>
<td>&gt;100,000 μs</td>
<td>&gt;100,000 μs</td>
<td>&gt;100,000 μs</td>
</tr>
<tr>
<td>From scratch power method</td>
<td>&gt;100,000 μs</td>
<td>&gt;100,000 μs</td>
<td>&gt;100,000 μs</td>
<td>&gt;100,000 μs</td>
</tr>
</tbody>
</table>


Environment: Intel Xeon E5-2690 2.90GHz CPU with 256GB memory
Experiments

Accuracy Comparison: Transition of Ave. $L_1$ Error

- This work
- Aggregation/Disaggregation [Chien et al.’04]
- Monte-Carlo [Bahmani et al.’10]
- Warm start (power method)
- From scratch (power method)

Comparable ($\sim 10^{-9}$) to naive methods

Environment: Intel Xeon E5-2690 2.90GHz CPU with 256GB memory
## Experiments

### Evaluation: Time & #Iter. for a Single-edge Insertion

| Dataset          | |V|   | |E|   | Max. Out-deg \(\Delta\) | Ave. Time  | Ave. #Iter. |
|------------------|----------------|-------|----------------|-------------|--------------|
| wiki-Talk [SNAP] | 2M            | 5M    | 100,022        | 589.6 μs    | 2.3          |
| web-Google [SNAP]| 1M            | 5M    | 3,444          | 7.2 μs      | 22.6         |
| as-Skitter [SNAP]| 2M            | 11M   | 35,387         | 288.4 μs    | 0.8          |
| FlickrTime [KONECT] | 2M         | 33M   | 26,367         | 95.3 μs     | 16.2         |
| WikipediaTime [KONECT] | 2M       | 40M   | 6,975          | 76.8 μs     | 46.0         |
| soc-LiveJournal1 [SNAP] | 5M       | 68M   | 20,292         | 17.9 μs     | 7.6          |
| twitter-2010 [LAW] | 42M      | 1,500M | 2,997,469      | 29,382.8 μs | 0.7          |
| uk-2007-05 [LAW]  | 105M        | 3,700M | 15,402         | 2.3 μs      | 0.0          |


Environment: Intel Xeon E5-2690 2.90GHz CPU with 256GB memory
Experiments

Evaluation: Relationship between $|E|$ & #Iter.

We plotted for 15 datasets

Ave. #iter. for edge insertion

Matches our theoretical result

Environment: Intel Xeon E5-2690 2.90GHz CPU with 256GB memory
### Experiments

#### Evaluation: Time & #Iter. for a Single-edge Insertion

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>wiki-Talk</td>
<td>[SNAP]</td>
<td></td>
<td></td>
<td></td>
<td>100,022</td>
<td>589.6 μs</td>
</tr>
<tr>
<td>web-Google</td>
<td>[SNAP]</td>
<td></td>
<td></td>
<td></td>
<td>3,444</td>
<td>7.2 μs</td>
</tr>
<tr>
<td>as-Skitter</td>
<td>[SNAP]</td>
<td></td>
<td></td>
<td></td>
<td>35,387</td>
<td>288.4 μs</td>
</tr>
<tr>
<td>FlickrTime</td>
<td>[KONECT]</td>
<td></td>
<td></td>
<td></td>
<td>26,367</td>
<td>95.3 μs</td>
</tr>
<tr>
<td>WikipediaTime</td>
<td>[KONECT]</td>
<td></td>
<td></td>
<td></td>
<td>6,975</td>
<td>76.8 μs</td>
</tr>
<tr>
<td>soc-LiveJournal1</td>
<td>[SNAP]</td>
<td></td>
<td></td>
<td></td>
<td>20,292</td>
<td>17.9 μs</td>
</tr>
<tr>
<td>twitter-2010</td>
<td>[LAW]</td>
<td></td>
<td></td>
<td></td>
<td>2,997,469</td>
<td>29,382.8 μs</td>
</tr>
<tr>
<td>uk-2007-05</td>
<td>[LAW]</td>
<td></td>
<td></td>
<td></td>
<td>15,402</td>
<td>2.3 μs</td>
</tr>
</tbody>
</table>


Environment: Intel Xeon E5-2690 2.90GHz CPU with 256GB memory
Experiments

Discussion: What is the Difference?

- twitter-2010 \((u, v)\) says “\(v\) follows \(u\)”
- uk-2007-05 \((u, v)\) says “\(u\) links to \(v\)”

\[ \text{# vertices s.t. out-deg} \geq k \]

Celebrities cause the performance degradation!!
Summary

Proposed an efficient & accurate method for **Personalized PageRank tracking** in evolving networks

**Theoretically**

Ave. $O(m + \Delta \log m / \epsilon)$ for $m$ edge insertions

**Experimentally**

Scales to a graph w/ 3.7B edges

**Future Work**

- Further Speed-up based on our observation
- Handle dangling nodes